Scaling the Namenode - Lessons Learnt

Dinesh Chitlangia, Manager @ Cloudera

September 21, 2021

About me

Dinesh Chitlangia

- Manager @ Cloudera
- Apache Ozone PMC/Committer
- Apache Hadoop Committer
- LinkedIn - Dinesh Chitlangia
- Github - dineshchitlangia
- Twitter - dineshneo
- Email - dineshc@apache.org

- Problems
- Causes
- Guidance

Commonly Reported Problems

- **Performance**
  - RPC Processing Time
  - GC pauses
  - Read/Write performance
  - Too long to start NN
- **Stability**
  - Frequent Failover
  - Frequent Crash

Common Causes

- Small files
- Sub optimal heap settings
- Missing RPC improvements
- Bad Applications / Mistuned Components
- Degraded Group Lookup
- Too frequent/delayed checkpointing
- Heavy Services co-located / Disk throughput
- Too much logging
- Degraded JN / communication between NN/JN/ZK

Guidance

- Optimize Logging
  - Disable getfileinfo audit logging
  - Async Logging
    - Namenode Audit
    - Namenode Edit
  - Reduce State Change Logging
    - BlockStateChange
    - StateChange

Guidance

- Optimize RPC
  - Service RPC & Handler count
  - RPC Congestion Control
  - RPC Call Queue
  - Datanode Lifeline Protocol

Guidance

- Dealing with Host Level issues
  - Dedicated disks for NN / JN
  - Don’t co-locate heavy services like HiveServer2, HBase Master/Region server, Yarn RM etc on the same host as NN
  - CPU Scaling should prefer Performance over Power Saving
  - Disable THP
  - Verify Kernel Configs after every OS patching

Guidance

- Group Lookup performance
  - Increase group cache timeout
  - Increase negative cache timeout
  - Add static mapping override for specific users

Guidance

- Failover / Startup
  - Optimal value for ZKFC timeout
  - Speed-up quota initialization
  - Throttle FSImage transfer bandwidth

- Bonus: If you do not have use cases which need to find last access time for files in HDFS, turn off access time precision to avoid NN writing an edit log entry.

Guidance

- Dealing with Heap
  - Each namespace object in NN is ~ 150 bytes
  - Block Size 128 MB
  - 128 MB File = 2 objects, 1 inode + 1 block ~ 300 bytes
  - But 128 1 MB files =256 objects,
    128 inodes + 128 blocks ~ 38.4K bytes
- 1 GB for every million blocks is a very conservative rule of thumb

Guidance

- Block Reports
  - Split Block Report by Volume
  - Reduce Full Block Report frequency
  - Batch incremental block reports

Guidance

- Manage External Factors
  - Control small files
  - Avoid **du**, use **ls** instead
  - Reduce Max Replication from default 512.
  - Hive/Tez - Merge map/reduce/intermediate output, ACID Compactions
  - Spark - Enable History Cleaner
  - YARN - Log aggregation
  - HBase - Carefully choose the RegionSplitPolicy, merge size 0 regions
  - Data retention policy

Reference

https://s.apache.org/NamenodeScalability

https://github.com/dineshchitlangia/NamenodeScalability

Questions?

Thank you!